NASA Grant NAG 2-123*

PILOT INTERACTION WITH AUTOMATED AIRBORNE DECISION MAKING SYSTEMS

Semiannual Progress Reports

August 1984 - February 1985
March 1985 - July 1985

William B. Rouse, Principal Investigator

John M. Hammer, Co-Principal Investigator

Christine M. Mitchell

Nancy M. Morris

C. Michael Lewis

Wan C. Yoon

Center for Man-Machine Systems Research

Georgia Institute of Technology

Atlanta, Georgia 30332

(NASA-CR-176986) PILOT INTERACTION WITH AUTOMATED AIRBORNE DECISION MAKING SYSTEMS Semiannual Progress Reports, Aug. - Jul. 1985 (Georgia Inst. of Tech.) 50 p CSCL 05H G3/54

N86-31218
THRU
N86-31220
Unclas
43136

*The NASA Technical Officer for this grant is Everett Palmer, NASA Ames Research Center, Moffett Field, CA 94035.

INTRODUCTION

This report covers progress during two six-month reporting periods: 1) August 1984 - February 1984, 2) March 1985 - July 1985. During these periods substantial progress has been made in three areas.

In the rule-based modeling area, Mike Lewis' Ph.D. work is nearing completion. This report includes two papers related to identification and significance testing of rule-based models, and a third paper on an application to CDTI data.

In the area of operator aiding, Wan Yoon's Ph.D. research is focusing on aiding operators in novel failure situations. Chris Mitchell has developed a discrete control modeling approach to aiding PLANT operators. Finally, Bill Rouse and Nancy Morris have developed a set of guidelines for implementing automation.

The third area of progress is the flight simulator hardware and software. While this development effort has taken much more time than originally envisioned, the hardware will be complete within two months and initial simulation software will then be integrated and tested.

SIGNIFICANCE TESTING OF RULE-BASED MODELS

The appended article by Lewis and Hammer describes how to test the significance of rules in rule-based models. The danger in rule-based model building is that the overall model may fit the data well but that individual rules may not contribute to this fit. The article explains several relatively easy methods for testing rule fit.

IDENTIFICATION OF RULE-BASED MODELS

The Human Factors and IEEE SMC conference papers on rule identification included with this report recapitulate much of the work in

the re-analysis of CDTI data [Palmer 1983] contained in our last report. Attention over the interim has largely focused on significance testing in the identification process. The SMC paper concentrates on methodological difficulties inherent in employing logical generalization and points out some of the strengths of alternate approaches.

The possibility of developing significance tests for logical generalization remains a paramount advantage over the top-down approaches. The Monte Carlo procedure described in the SMC and Human Factors papers proved effective but inefficient. Running in the background at low priority it has taken about 10 minutes per iteration. One thousand iterations are used. The working paper on representation describes well formed formulas in VL1 which might be used in deriving a closed form significance test. If attainable it would avoid the inefficiencies of repeated search by generating counts of rules directly.

The primary effort in rule identification is now being directed toward the PLANT [Morris 1983] data. Preliminary identification of rules has indicated a lack of stationarity associated with shifts in operator goals and phases of operation. This result is not unexpected as task constraints led KARL [Knaeuper 1983], a production system model of the operator, to employ explicit state->state rules to achieve such transitions.

Since these shifts are unobservable, state vectors will be augmented with an oracle variable encoding shifts KARL "would have made". Shifts in phase dictated by a discrete control model of PLANT [Mitchell 1984] will also be employed in a parallel effort.

AIDING THE OPERATOR DURING NOVEL FAULT DIAGNOSIS

An aid containing a qualitative device model and designed to counteract human decision-making biases is being investigated. This aid is designed to be used during novel failures, which are defined as failures that are not covered by the operator's training or procedures. The remainder of this section covers the qualitative model, decision aiding, and the applicability of the aid. Further details are in an appended paper.

A qualitative model [deKleer and Brown, Davis, Forbus] represents a physical device as a network of components and connections. Each component and connection can have several discrete states. The behavior of a component (in terms of connection flows such as current) is governed by rules. Component state transition is also governed by transition rules. A solution to a diagnostic problem is an assignment of states to components and connections that explains the observed symptoms and obeys physics as described by the rules. The qualitative model is included in the aid to assist the human in reasoning about the physical device.

The aid also is designed to counteract some human decision making biases. This aiding takes a number of forms. Working memory is augmented with the display of hypotheses and data. The human tendency to forget or overlook is counteracted by the aid's mechanistic reasoning.

Applicability

The applicability of the kind of aid is whenever a novel failure occurs. In commercial aviation, such failures are relatively rare. They would seem to be more common in process control and space flight. The most applicable area would seem to be commercial space loads. Because

most of these are one shot, non-life threatening tasks, the operator's training will be limited than on the operation of spacecraft itself. The economic consequences are high for an improperly diagnosed payload problem.

A DISCRETE CONTROL MODEL OF PLANT

The appended working paper by Chris Mitchell develops a discrete control model of the PLANT process control simulation and discusses the potential use of the model as a basis for a new human-computer interface for PLANT.

GUIDELINES FOR IMPLEMENTING AUTOMATION

The appended paper by Bill Rouse and Nancy Morris summarizes recent efforts to understand how people perceive automation and the influence of these perceptions on acceptance of automation. A set of eight guidelines are proposed as a possible means of enhancing acceptance.

FLIGHT SIMULATOR HARDWARE AND SOFTWARE

The hardware is scheduled for completion by September 15, 1985. The following have been completed.

1. Wiring to the sensors
2. Pedestal has been reinstalled.
3. Keyboards are mounted in the pedestal.
4. The CRT's have all been tested and mounted.
5. A force feel system with trim for the elevators was designed and installed.

Progress on the software is as follows.

1. An engine display program has been completed.
2. A very simple flight instrument display has been completed.
3. The existing simulation has been revised to run under UNIX.

The hardware changes that remain are:

1. Cooling fans must be installed for the CRT's.
2. The glare shield must be installed.
3. Some metal panels must be added to shroud CRT's and otherwise close up the cockpit.

The software that remains to be done can be categorized as follows.

1. Modifying the simulation to accept inputs from the A/D converter.
2. Modifying the simulation program to drive different, multiple displays for the instruments.
3. Integration - simply making sure that everything is connected and works the way it is supposed to.

The completion of the above will demonstrate that the simulator will work.

The simulator has come along at a much slower rate than anticipated. We did not initially realize the complexity of the project. As compensation, the final cost of the simulator will be a small fraction of the cost of a new one.

REFERENCES

1. Davis, R. "Reasoning from first principles in electronic troubleshooting," Int'l. J. Man-Machine Studies, V19, 1983.

2. deKleer, J. and Brown, J.S., "Assumptions and ambiguities in mechanistic mental models," in Gentner and Stevens (eds.), Mental Models, Hillsdale, NJ: Lawrence Erlbaum, 1983.

3. Forbus, K.D., "Qualitative reasoning about space and motion," in Gentner and Stevens (eds.), Mental Models, Hillsdale, NJ: Lawrence Erlbaum, 1983.

4. Knaeuper, Annette. A model of human problem solving in dynamic environments. M.S. Thesis. Georgia Institute of Technology, 1983.

5. Michalski, R.S. Knowledge repair mechanisms: evolution vs. revolution. Proceedings of the Third International Machine Learning Workshop. New Brunswick, N.J.: Rutgers University, 116-119, 1985a.

6. Michalski, R.S. and Winston, P.H. Variable Precision Logic, A.I. Memo. Boston, Mass.: Artificial Intelligence Laboratory, M.I.T., 1985b.

7. Michalski, R.S. and Reinke, R. Incremental Learning of Concept Descriptions, Machine Intelligence 11, J.E. Hayes, D. Michie, and J. Richards, eds., Oxford: Oxford University Press, 1985c.

8. Mitchell, Christine M. A discrete control model of PLANT, working paper. Center for Man-Machine Systems Research, Georgia Institute of Technology, 1984.

9. Morris, Nancy M. Human problem solving in process control, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1983.

10. Palmer, Everett. "Conflict resolution maneuvers during near miss encounters with cockpit traffic displays", Proceedings of the Human Factors Society-27th Annual Meeting, Santa Monica, CA.: Human Factors Society, 757-761, 1983.

APPENDIX

REPRESENTATION for CLOSED FORM SIGNIFICANCE TESTING in VL1

C. Michael Lewis
Center for Man-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology

(A Working Paper)

Programs for machine learning and systems for logical induction such as that of Carnap share certain characteristics:

> A language, L, consisting of a vocabulary of predicates and connectives is defined for describing a set of observations. Induction is achieved by identifying generalized statements (wff's) describing the observations. In this paper these statements will be referred to as "rules".

As relations among predicates are restricted to legal connectives (usually: &, V, ~). inductive inference, based on the coincidence of predicates rather than the correlation of values, results. This distinction is basic to the problem of inference since relations are identified between logical phrases and their consequents rather than between individual "variables".

In machine learning and other applications of inductive logic this usually reduces to a discriminant procedure. Predicates are divided into a group describing the observations and a class of mutually exclusive discriminant predicates one of which is designated as the positive case, '+', while the others are treated as negative cases. When used to describe behavioral 'rules', phrases describing the stimuli which are accompanied by a particular response are discriminated from all others.

While Carnap's system, based on degrees of confirmation, allows rules to describe negative instances, machine

learning programs typically inspect only valid rules to restrict search.

If this logical generalization is to be used to form inferences from actual observations the relationship between the data and its population must be considered. The primary threat to validity in the identification of rules through combinatorial coincidence lies in the ability of such procedures to identify apparent consistencies (in the sample) where none exist (in the population). This occurs because the generalization procedure can examine the data in so many ways that it is possible to discover a relationship that is only due to chance. To guage our confidence in a particular rule, it is necessary to find some benchmark with which it can be referenced. A logical choice is the situation in which there is no relation between observation descriptions and their classification as '+' or '-,' then any pairing of observation descriptions and discriminant classifications would be equally likely (principle of indifference). If there are N observations and k of these observations are classified as '+' then there are N!/(N-k)!k! distinct mappings from the observation descriptions to the responses. These mappings will be referred to as instantiations.

The rub here is that the possible instantiations need to be expressed in terms of the rules which would be identified rather than the mappings of observation descriptions to responses. themselves.

The GENERAL PROBLEM

Constraints:

1- The syntax of language, L, which defines the ways in which predicates & connectives can be combined to form rules (wff's)

2- The set of observation descriptions which determines the discriminations which can be made

3- The number of observations classified '+'

The Problem:

Devise a method for determining the relative frequency over instantiations with which a rule of equal or greater generality would be identified. Generality, here, is defined as the number of observations described by a rule.

My Problem:

I am using a covering algorithm (Aq, Michalski 1973) to identify pilot strategy (system of rules) in maneuvering to avoid intruders. The Aq algorithm identifies rules in the VL1 (Michalski's terminology) language, a simplified propositional logic. VL1 allows only one predicate per class of predicates to be true in a particular instantiation. VL1 syntax allows only one place predicates. example: black(bird). Disjunction is only allowed within classes of predicates. example: black(bird) V blue(bird). Conjunction is only allowed between classes of predicates. exam-

ple: black(bird) & is-raven(bird).

For notational convenience Michalski defines a new entity, a selector, to describe classes of predicates. A selector names the predicate class and lists the predicates involved in a disjunction. example: [color-bird=black V blue]. Redefined in terms of selectors, VL1 allows only conjunction of selectors. Since generalization of the observation descriptions is performed to discriminate the '+' classification, this relationship is represented as An example of a wff in VL1 is:

[x1=a V b] & [x2=d V e]-> F

This rule says that if the predicate evaluating true for class x1 is a or b and class x2 is d or e then F(observation) is true.

An advantage of VL1 for devising a significance test is that this syntax is highly restrictive making the number of possible rules to be considered relatively small. For example a rule to cover the observation descriptions:

[x1=1] & [x2=1] & [x3=1]

[x1=1] & [x2=2] & [x3=1]

[x1=2] & [x2=1] & [x3=1]

MUST ALSO INCLUDE

[x1=2] & [x2=2] & [x3=1]

To produce a generalized conjunction of selectors

[x1=1 V 2] & [x2=1 V 2] & [x3=1]

I will refer to this syntactic property as a rectangularity constraint.

Despite the simplicity of the VL1 language and its restrictive syntax there are a very large number of rules which would be identified across the instantiations.

A Tentative Representation:

1- A Monte Carlo approach of repeatedly running the identification program on randomly selected instantiations is inelegant. It also unnecessarily expends resources on instantiations in which rules, less general than those being tested are identified.

2- A solution may be to ennumerate instantiations for which rules as/or more general would be identified had the identification program been run. Since only rules of substantial generality should be objects of testing, reduction in generation should be achieved. A threshold could be set so that the program terminates after ennumerating > N instantiations corresponding to a predetermined significance level, introducing further

economy.

Ennumerating Instantiations

The no relation benchmark corresponds to the distribution of the most general rule across instantiations. Since for any particular rule to be identified all observations must belong to the K observations in the '+' class, let k= the number of observations covered by a particular rule.

Eq. 1

(N-k)!/(N-K)!(K-k)! is then the number of instantiations in which that rule is valid.

Obtaining the set of possible rules of generality >= that being tested must be considered. The notion of discriminant equivalences will be introduced to accomplish this.

Definition: Discrimination level- a particular predicate class or conjunction of predicate classes. For example [x1=a,b....] would be a selector for a predicate class,
[x1=a,b,...] & [x2=c,d....] would be a selector for a conjunction of predicate classes.

Definition: Discriminant equivalence

Let M be a discrimination level. A discriminant equivalence, dq(M,i) in level, M, is a set of observation descriptions having identical predicate or conjunction of predicates

for the predicate or predicate classes of M.

Due to the syntax of VL1 (conjunction of selectors), rules can only be formed at a single level if the degenerate case of a selector specifying all members of a predicate class is excluded. As a consequence any rule at level M must either include or exclude all members of a dq(M.i) at level M and the dq(M.i)'s at each level represent a complete partitioning of the observation descriptions. As a consequence:

1- discriminations made by any rule at level, M. may be represented as a conjunction of dq(M.i)'s at level M.
2- All discriminations among observations in VL1 are covered if all levels are represented in this way

Since knowing the number of observation descriptions described by a rule allows the ennumeration of instantiations for which it is valid. (Eq.1). this provides a basis for forming and counting rules in accordance to generality across instantiations.

We, however, are interested in counting <u>instantiations</u> for which rules of >= generality than that being tested would be identified. Since it is possible for a rule to be

valid yet not be the most general identifiable rule for a particular instantiation, the ennumeration must be adjusted. A rule of greater generality will be said to dominate a rule of lesser generality for instantiations for which both are valid. For example: any rule at the same level which includes an additional dq(M.i) will dominate. Example: [x1=1,2] & [x2=2] dominates [x1=2] & [x2=2], the dq(M.i)'s in this instance are observation descriptions for which x1=1 & x2=2 and observation descriptions for which x1=2 & x2=2.

Between Level Dominance

Definition: Dominance set, Dq(L,M,i)

A dominance set, Dq(L,M.i), at level L of dq(M.i) at level M is defined as the minimal set of dq(M.i)'s at level M that contains all of the observation descriptions contained by dq(M,i) and is of cardinality < K, the number of observations in the '+' class

Let N= number of observation descriptions

K= number of observations in class '+'

|dq(M,i)|=k

|Dq(L,M,i)|=k*

THEN

$$(N-k)!/(N-K)!(K-k)! - (N-k^*)!/(N-K)!(N-k^*)!$$

is the number of instantiations in which dq(M.i) is not dominated by any rule from level M.

Minimal dominance rules at level, L, for rules at level, M, (conjunctions of dq(M.i)'s) are then simply the union of the corresponding Dq(L,M,i)'s at level L (with additional dq(L,i)'s at level L included as required by rectangularity constraints). Undominated instantiations can be ennumerated in the same way as before.

To find the undominated instantiations of a rule across all levels, however, requires consideration of co-dominance and multiple dominances as well.

example:

A rule at level L is dominated by its corresponding minimal dominance rule at level M. This rule may in term be dominated by its own minimal dominance rule at level N... This problem can be dealt with without recursion by considering all unions for a set of minimal dominance rules (discarding those > K) for the initial rule at level, L, and applying the inclusion/exclusion principle.

To use this representation in deriving a significance test for VL1 rules will require:

1- Some way to ennumerate instantiations for >= rules directly for the dq(M.i)'s and Dq(L,M.i)'s without recourse to actually

forming the rules (tagged generating functions?)

failing this

2- Some computationally cheap way to find the undominated instantiations of a rule without having to consider all unions of its minimal dominance rules

A Discrete Control Model of PLANT

Christine M. Mitchell

Supported by NASA-Ames Grant #NAG-2-123

December 1984

Introduction

This task entailed the development of a model of the PLANT system using the discrete control modeling techniques developed by Miller (1985). Discrete control models attempt to represent in a mathematical form how a human operator might decompose a complex system into simpler parts and how s/he coordinates control actions and system configuration so that acceptable overall system performance is achieved. Basic questions include knowledge representation, information flow, and decision making in complex systems. The structure of the model is a general hierarchical/heterarchical scheme which structurally accounts for coordination and dynamic focus of attention. Mathematically, the discrete control model is defined in terms of a network of finite state systems.

The discrete control model can be thought of as a possible representation of an operator's internal model of the system plus a control structure which specifies how the model is used to solve the decision problems which make up the control functions. Specifically, the discrete control model accounts for how specific control actions are selected from information about the controlled system, the environment, and the context of the situation. The objective is to provide a plausible and empirically testable accounting and, if possible, explanation of control behavior.

Theoretical details and practical mechanics of discrete control modeling are detailed in Miller (1985) and Mitchell (1980). The model described below assumes most of this material as background.

Model Preliminaries

The first step in constructing a discrete control model is to specify the lowest level of description, the output of the model and the bottom nodes in the finite state network. Several discrete control models have based their structure on configurations of system switches (Miller, 1979; Mitchell, 1980). A model like this for PLANT, for example, would utilize the configuration of valves and the flow of resources (i.e., PI and PO) through the system. The initial model development for PLANT in fact began at this point. As modeling progressed, however, it became clear that the more interesting output of a PLANT discrete control model was the operator commands which were employed to configure and optimize PLANT. The commands available to the operator are summarized in table 1.

Using operator commands as the lowest level finite state nodes of the discrete control network, the model attempts to explain an operator's choice of commands based on system state and current operator function or procedure. The model is normative in that it is constructed using both the rules of the system as specified in PLANT documentation and the procedures provided to PLANT operators (Morris, 1983). Such a model could be used to "explain" operator behavior, that is, to justify and contextualize a sequence of operator actions based on goals and objectives. In addition, a normative discrete control model may be useful in designing an adaptive user interface that is responsive to user needs. Additional details on applications of the PLANT discrete control model follow the presentation of the model itself.

ovI,J	Open the valve between tanks I and J
cvI,J	Close the valve between tanks I and J
ocK	Open one valve per tank in column K
ccK	Close one valve per tank in column K
otI	Open all valves from tank I
ctI	Close all valves from tank I
piN	Set input per input tank to N units
poN	Set output per output tank to N units
skN	Skip N iterations; the system will be updated N times before the display is updated
flI,J	Check the flow from tank I to tank J
afI	Check all flows from tank I
rvI,J	Repair the valve between tanks I and J
rpI	Repair the pump associated with tank I
rtI	Repair the rupture of tank I
rs	Repair the PLANT safety system
st	System trip; close all valves and stop all input and output

Table 1. PLANT Operator Commands

The PLANT Control Network

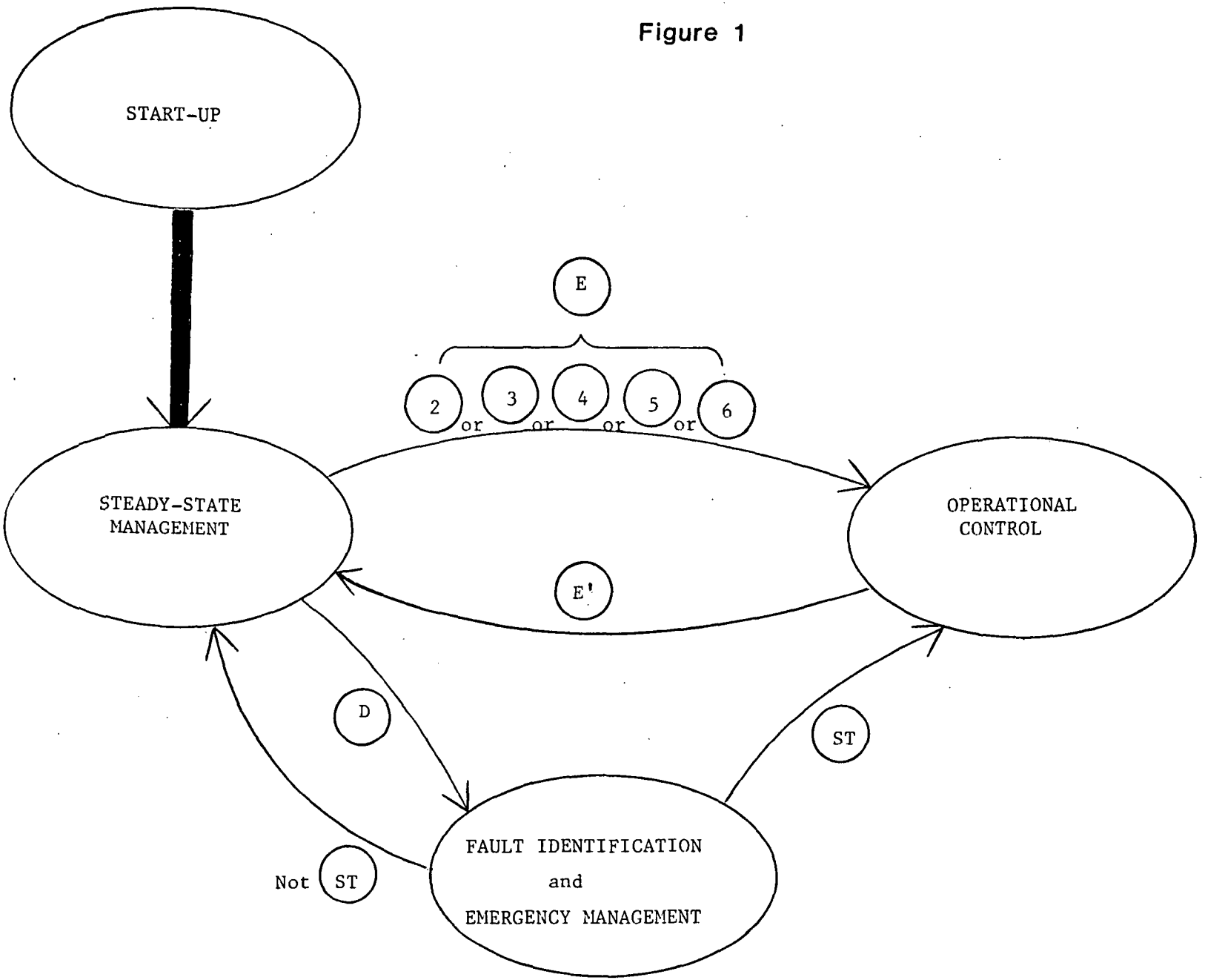
At the highest level, there are four major operator control functions for PLANT. As depicted in figure 1, the first control function that an operator engages in is system start-up. Once this set of activities is completed the operator unconditionally transitions to the function of steady-state management. For PLANT, steady-state management is a monitoring and fault detection state. In this state, the operator is essentially a supervisory controller, watching a fairly autonomous system operate within the boundaries specified by the system configuration.

From steady-state management, an operator can, under certain conditions, transition to a more active control function. If a fault is suspected or if the system is approaching an out-of-control condition, the operator engages is fault identification or emergency management. On the other hand, while engaged in steady-state management the operator may notice the evolution of one or more symptoms which suggest the need for proceduralized manual control.

Figure 1 and the associated footnotes depict these transitions. Several conventions are used in this figure. The heavy black arc between the startup node and the steady-state management node is used to denote an unconditional transition. This convention will be used both in figure 1 and in subsequent figures. Conditional arcs are those that denote state transitions which occur only when enabling conditions are met. For example, steady-state management shifts to operational control when one of the conditions calling for a procedure are met.

PLANT - Control Overview

Figure 1



Control Overview Notes & Symbols

1. The heavy black arrow from start-up to steady-state management denotes an automatic or unconditional transition, i.e., once the start-up control function terminates there is an automatic transition to the steady-state management function. Moreover, as the figure suggests, the start-up function is performed exactly once for the control session.

2. Control shifts from start-up to steady-state management and then to fault identification and emergency management when a fault or operational problem is suspected.

   (D) : fault suspected due to
   - random tank check
   - drop in resources
   - insufficient number of system trips
   - unmanageable number of system trips

   (E) or (i) : symptoms are present which call for the use of procedure i, i = 2,3,4,5,6

3. Control shifts from operational problem solving to steady-state management when system is reconfigured at a minimum stabilization (E'). (E') will be further specified in the operational control function.

4. (ST) indicates an operator-initiated system trip.

The conditional arcs at this level of the network are straightforward. Condition (E) is an enabling condition which is true when the system state requires the use of a prespecified procedure. Condition (E') is a minimum level of system stabilization; this condition is set to true at the completion of each procedure in operational control. Condition (D) is true when a system fault is suspected during routine management or if the number of system trips is so excessive as to make the operator feel that PLANT is in an out-of-control condition. Finally, condition (ST) indicates that the operator initiated a system trip.

The structure of discrete control models is both hierarchic and heterarchic. The portion of the model depicted in figure 1 is at the highest level of the hierarchy and depicts the somewhat heterarchic activities that take place at this level. For the PLANT model, the next level of detail explores particular activities or subfunctions within each of the major control activities previously described. Figures 2-9 constitute the remainder of the model.

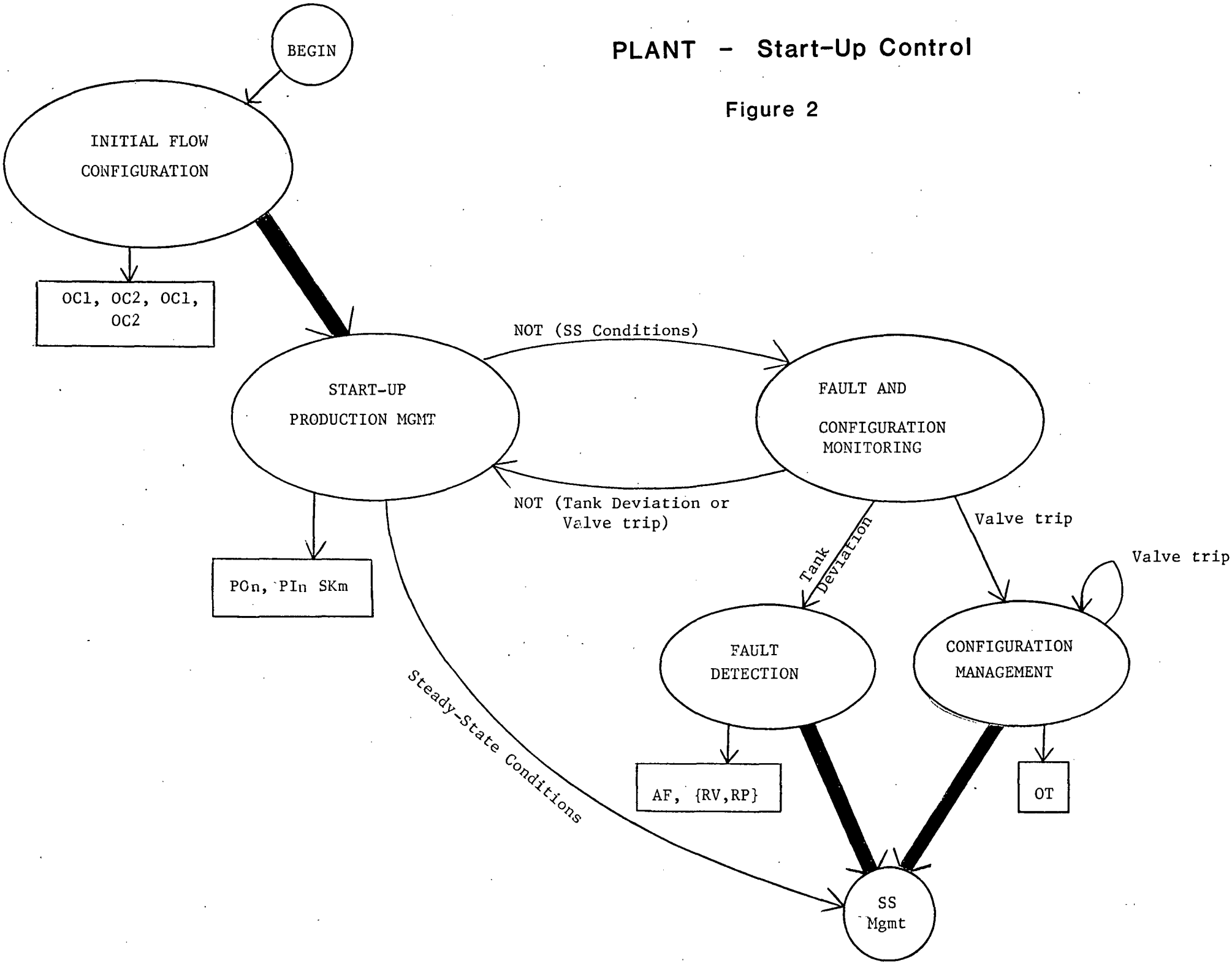
Start-Up Control

The start-up control function (figure 2) consists of three major subfunctions. Initially, the operator opens the valves among the tanks with a sequence of commands that completes the initial flow configuration.

The conventions used in figures 2-11 include the use of elliptical nodes for operator functions or actions and rectangular boxes for commands. The distinction being made is one of activity versus intention. The commands are activities undertaken to accomplish an objective of a function.

PLANT - Start-Up Control

Figure 2



Start-Up Control Notes

1. Steady-state conditions: pi = po = 230

2. Tank deviation: inequality of tank heights within columns

3. Start-up production management: n = 50,100,150,200,210,220,230;
   m = 5

Once the initial flow configuration is completed the operator unconditionally transfers to a start-up production management function, the function which increases input and output to a steady-state condition. Each start-up production management task increases output (PO), input (PI), and may skip (SK) one or more PLANT iterations. Once a round is accomplished the operator checks to see if steady-state conditions have been met, i.e., PI = PO = 230. If so, the start-up function is concluded and the operator unconditionally transitions from start-up control to steady-state management. If start-up conditions have not been met, the operator transitions to a fault and configuration monitoring subfunction, in which tank heights are scanned to ensure levels are the same within columns and no valve has tripped. If a tank deviation or valve trip has occurred the operator performs the appropriate remedial action, i.e., opens the tripped valves or checks flows to identify and fix the failed valve or pump. Once a fault or valve trip has been identified the start-up control function concludes and the operator transitions to the steady-state management function. If after checking tank levels and valves, no problems are detected the operator again commences on another start-up production management subfunction.

To summarize, the start-up control function terminates in one of two ways. Typically, start-up control is completed when the operator has configured PLANT into a minimally stable and optimal mode. If, however, a fault is detected before start-up is complete, diagnostic and compensation procedures are performed and the operator terminates start-up control and engages in steady-state management.

Steady-State Management

Following the completion of start-up control, steady-state management is the next high level control function undertaken (figure 3). This control function has three major components: monitoring and fault detection, configuration management, and production optimizaton.

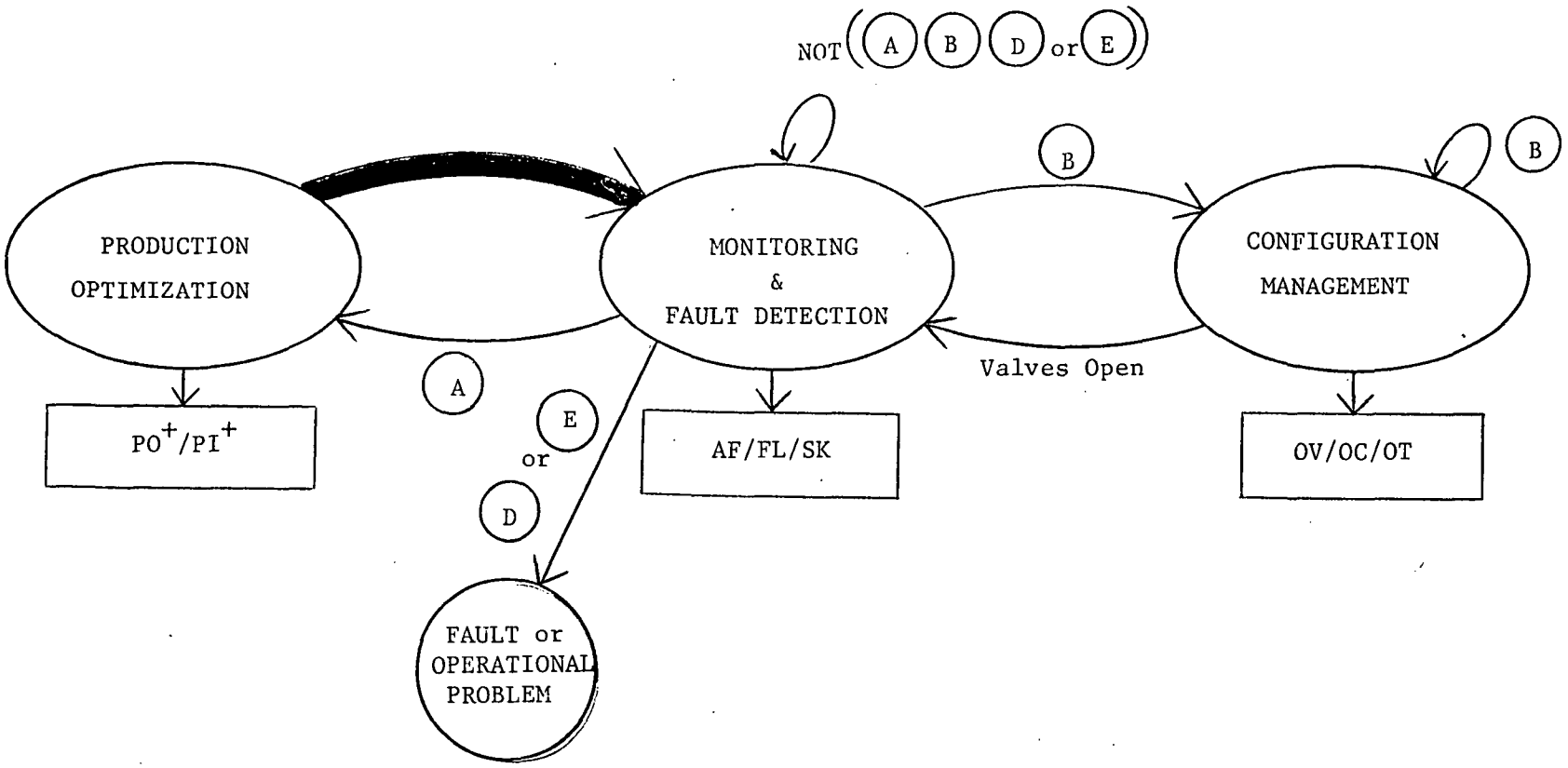
Monitoring and fault detection is the central subfunction. An operator may perform this function with visual scans of the PLANT displays together with iteration skips (SK) or by actively testing tank flows (AF,FL). The operator terminates this function to undertake another steady-state management subfunction if one or more valves trip, if a repair crew completion message arrives, or if configuration is acceptable and production optimization is required. Configuration management is the subfunction which keeps all valves open under normal conditions and reopens valves after repair completion. Product optimization entails one or more commands to balance and/or increase input and output. This subfunction is pursued when the PLANT is stable and production is less than the specified goal, i.e., PO = PI = 230. Operator control remains in the steady-state management function, alternating among its subfunctions, until some problem arises. Operator control leaves the steady-state management function when a problem requiring either proceduralized control or fault detection occurs.

Operational Control

Operational control is an operator function which is, as compared to steady-state management, proceduralized and low level; in this function, the operator exercises a great deal of direct manipulation over the system (figure 4). Depending on the symptoms, the operator engages in

PLANT - Steady-State Management

Figure 3



(A) Input/Output Tuning Required (pi ≠ po or pi < 230 or po < 230) and low frequency of valve trips and tanks are within acceptable range*)

(B) Isolated valve trip or repair crew completion

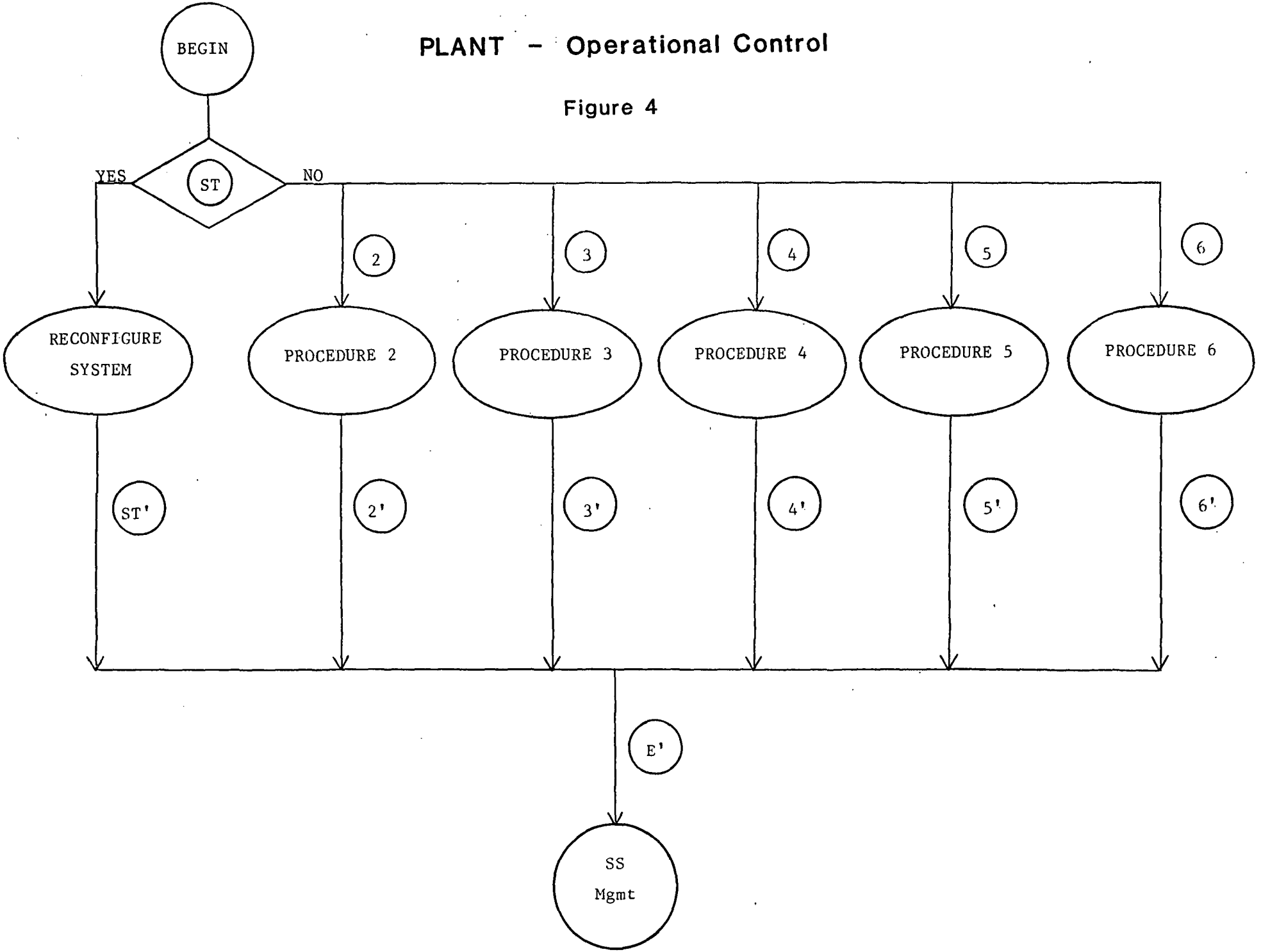
D Fault suspected (see control overview)

E Symptoms of need for proceduralized operation (see control overview)

* largest difference < 30 units, all levels are > 10 and < 90 units

PLANT - Operational Control

Figure 4



Operational Control Notes

1. (ST) indicates an operator-initiated system trip

2. (i) symptoms present which require use of procedure i, i = 2,...,6.

3. (ST') or (i') conditions have been met which terminate the procedure, control then shifts back to steady state management.

4. (E') system stabilized E' = T if (ST') or (i') for i = 2,3,4,5,6

one of the prespecified control procedures (see Morris, 1983) or in reconfiguration after a system trip. When an operational control procedure ends, operator control returns to steady-state management after a minimum system stabilization point is reached.

The simplest subfunction in operational control is PLANT reconfiguration after a system trip (figure 5). This simply consists of reopening all valves. Once all valves are opened, operator function transitions back to steady-state management. Input and output production increases are handled by the product optimization subfunction of steady-state management and are accompanied by fault detection and monitoring.

The remaining procedures in the operational control function are similar to those used by Morris (1983) to train PLANT controllers. The symptoms that indicate a need for these procedures are summarized in Appendix A. The PLANT discrete control model has structured the steps in the procedures and, as a result, they are not quite isomorphic to those used in operator training. One major difference betwen Morris' procedures and the model's procedures are the termination points of the procedures. The model's procedures terminate as soon as a minimal point of system stabilization is reached; system optimization, input and output increase for example, are completed as part of the steady-state management function. The result of these changes in terms of operator control activities, however, should be equivalent. The model's version of these procedures is summarized below.

Procedure 2 addresses the problem arising when PLANT's input column tank levels are higher than those of the other columns (figure 6). This procedure consists of three subfunctions: production limitation wherein, depending on system symptoms, input and output are reduced; valve

PLANT - Operational Control

Reconfigure System After Operator-Initiated System Trip

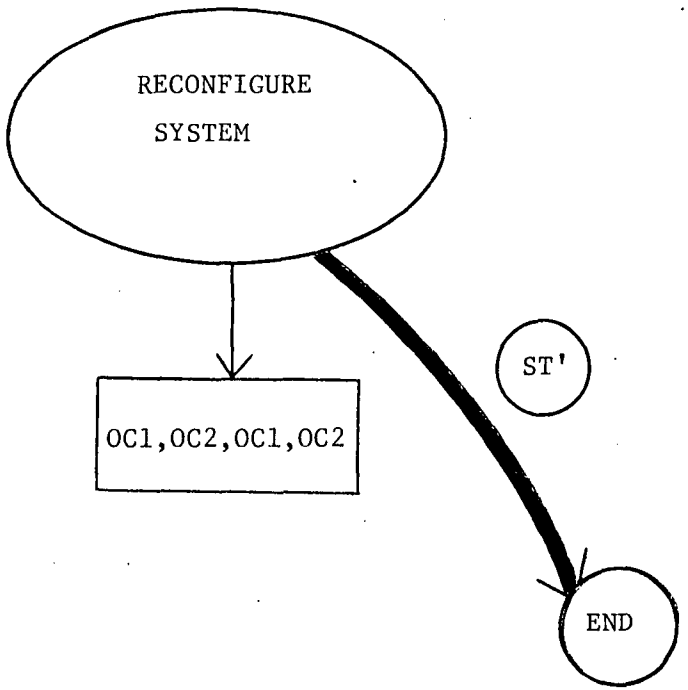
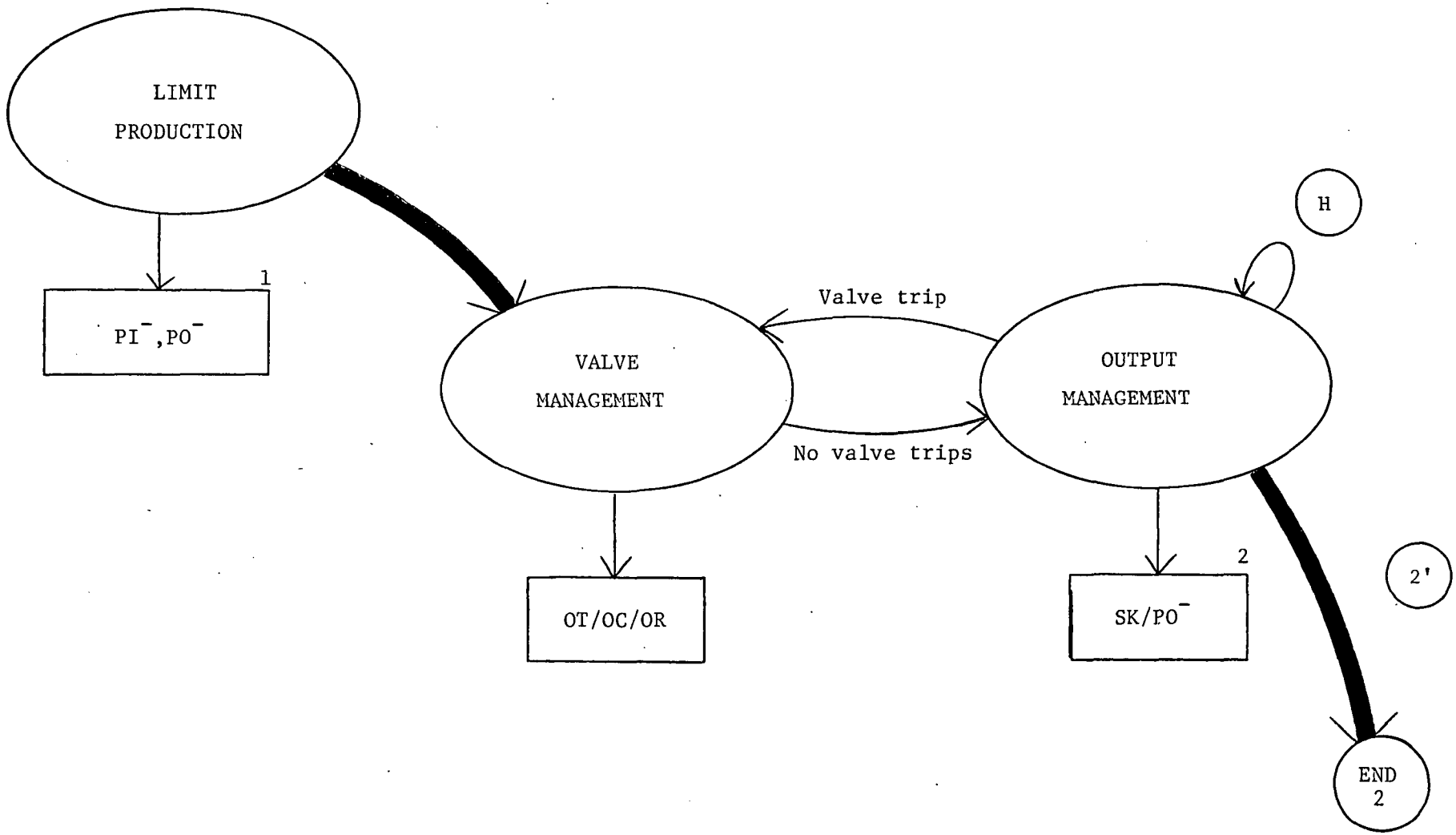


Figure 5

PLANT - Operational Control

Procedure 2 BC Problem for Input Column

Figure 6



PLANT-Operational Control Procedure 2 Notes

1. Rule IF any tank in Col. 1 > 90
   THEN PI: = 0 & PO: = (0 - 25)
   ELSE PI: = 50 & PO: = (50 - 75)

2. Rule IF tanks in Col. 2 and 3 continue to drop and iteration = 10
   THEN PO: = Ø

(H). PO > Ø AND less than ten iterations

(2'). PO: = Ø OR iterations > 10

management which keeps tripped valves open; and output management which monitors output and adjusts production to keep columns 2 and 3 fairly balanced. The procedure concludes when columns 2 and 3 stabilize or if production output is set to zero.

Procedure 3, a condition in which column 3 tank levels are too low, also begins with a production curtailment function and unconditionally transitions to a valve management state (figure 7). From there, the operator monitors valve trips, input, and output until columns all stabilize (i.e., columns 1 and 2 do not continue to increase and column one does not continue to decrease) or until both input and output are set to zero.

Procedure 4 addresses the problems of high tank levels in column 1 and low levels in column 3, problems which combine those of procedures 2 and 3 (figure 8). The steps required in procedure 4 are very simple; first input is reduced, then output is reduced.

Procedures 5 and 6 (figures 9 and 10) address imbalances within columns rather than the imbalance between columns addressed in procedures 2 through 4. Since within column imbalances are often due to system faults, an initial concern is to examine the system for component failures. Procedure 5 first limits input, then examines the high tank for possible valve failure. Following failure detection, the operator also limits output. Finally, the operator engages in a set of monitoring and tuning activities to compensate for the imbalance and/or component failures. Tripped valves are opened. If column 1 shows an excessive imbalance among tanks, valves are temporarily closed. Similarly, if tank levels in columns 2 and 3 continue to decrease over time, output is

PLANT - Operational Control

Procedure 3: BC Problem for Output Column

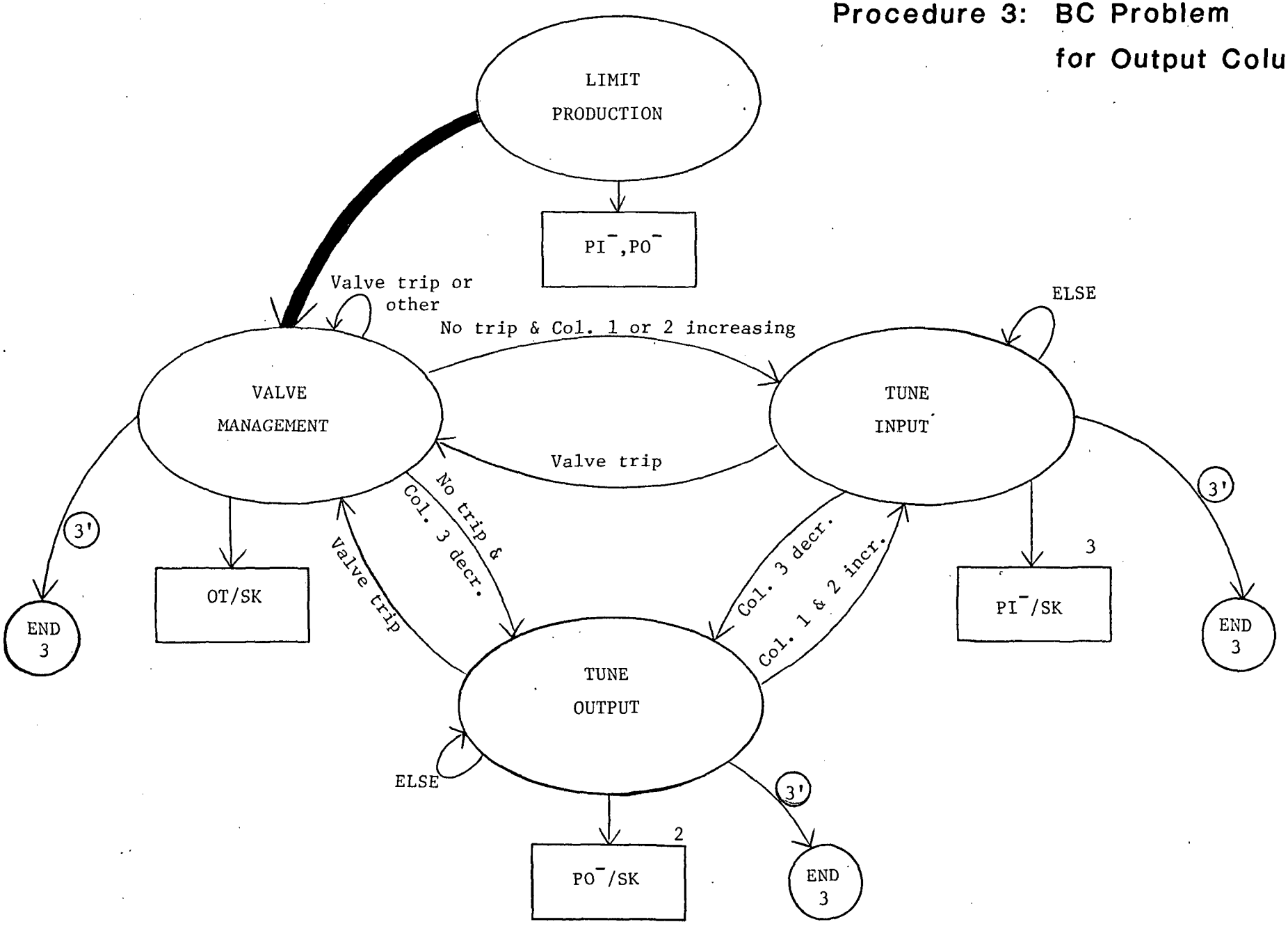


Figure 7

PLANT-Operational Control Procedure 3 Notes

1. PI: = (50 - 100); PO: = 50

2. IF Col. 3 < 5 then PO: = Ø
   ELSE IF Col. 3 < 10 then PO: = 25

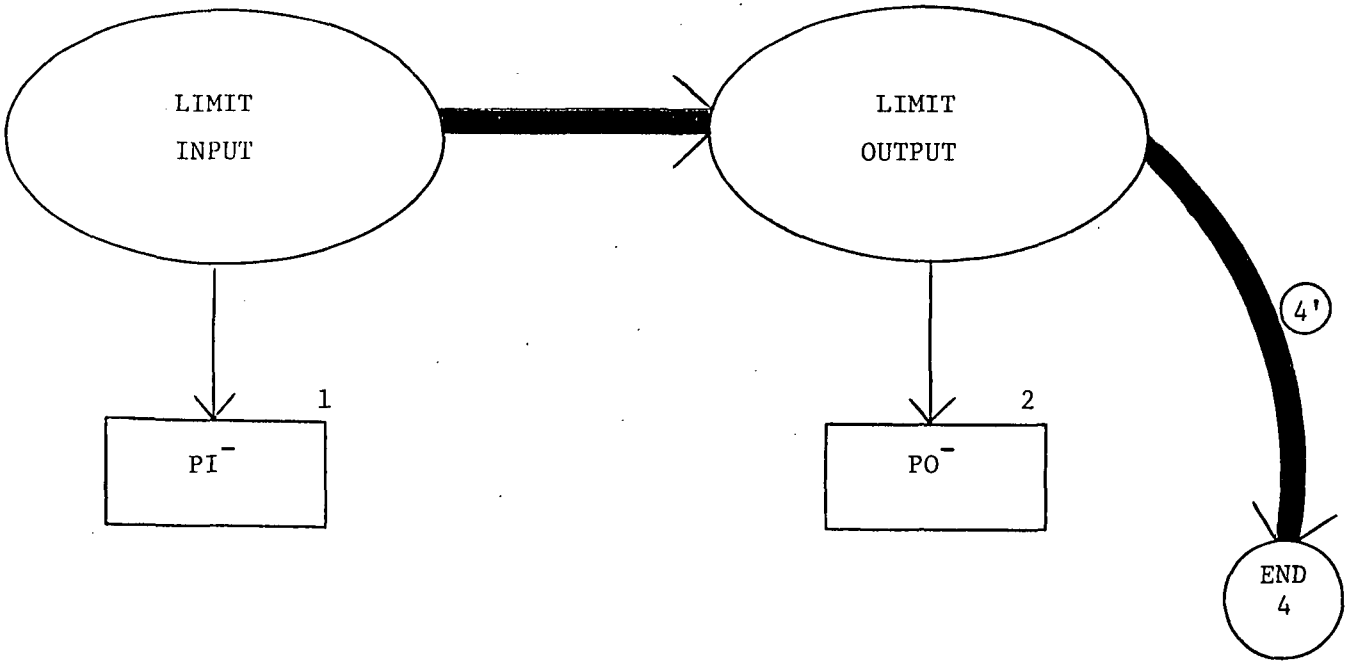
3. IF Col. 1 or 2 increases over 5 iterations, PI: = (25 - 50):
   IF frequent valve trips, PI: = 0

(3'). IF PI = PO = Ø or Col. 1 and 2 and 3 are stabilized

PLANT - Operational Control

Procedure 4: BC Problem for Input and Output

Figure 8



1 Rule: IF col. 1 > 90 then PI: = ∅
ELSE PI: = 50

2 Rule: IF col. 3 < 5 then PO: = ∅
ELSE PO: = 50

Figure 9

PLANT - Operational Control

Procedure 5: WC Problem for Input Column

1 Rule: IF Col. 1 tank > 90 then PI: = Ø
ELSE PI: = 50

2 Rule: PO: = (50 - 150)

3 Rule: PO: = (0 - 50)

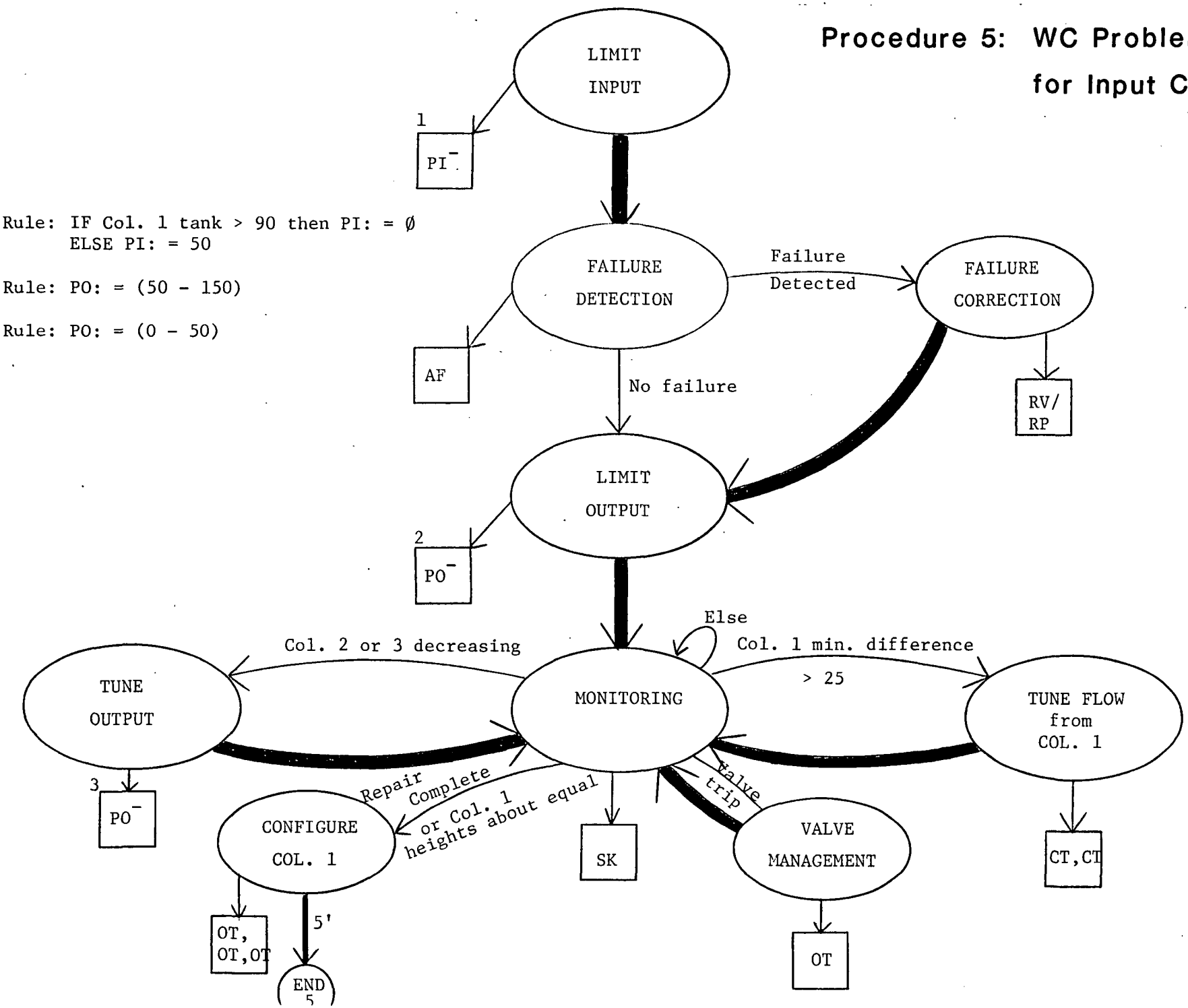
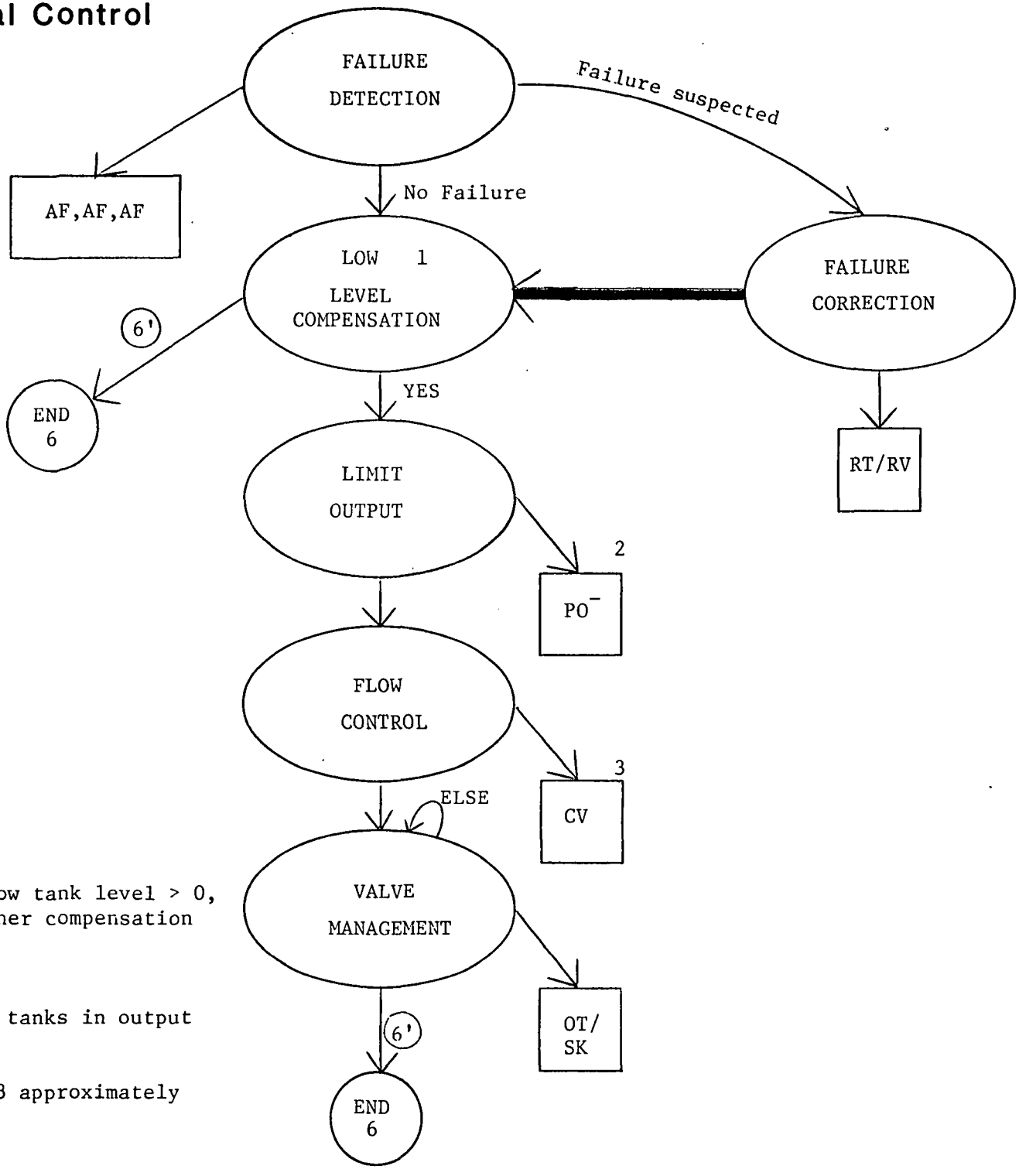


Figure 10

PLANT - Operational Control

Procedure 6:

WC Problem in Output Column



1: Decision Rule: If low tank level > 0, further compensation

2: PO: = (0 - 50)

3: Close valves to high tanks in output column

(6'): All tanks in column 3 approximately the same height

reduced. These activities continue until the system stabilizes or the repair on the failed component is completed.

Procedure 6 begins with a failure detection and, if necessary, compensation subfunction; then, a decision is made about whether compensation is needed. If not the procedure is ended. If more intervention is required, output is limited, flow to high output tanks is restricted, and tripped valves are opened until tank heights in column 3 stabilize.

Fault Identification and Emergency Management

The final high level control function is fault identification and emergency management (figure 11). This control function is always reached from steady-state management and the transition occurs due either to a suspected or detected fault or to a loss-of-control feeling on the part of the operator.

The system trip subfunction is the least straightforward. The option is invoked when the system is so unstable that the operator feels a total loss of control. It might be argued that this procedure is never really required or justified; a competent operator has less catastrophic procedures available.

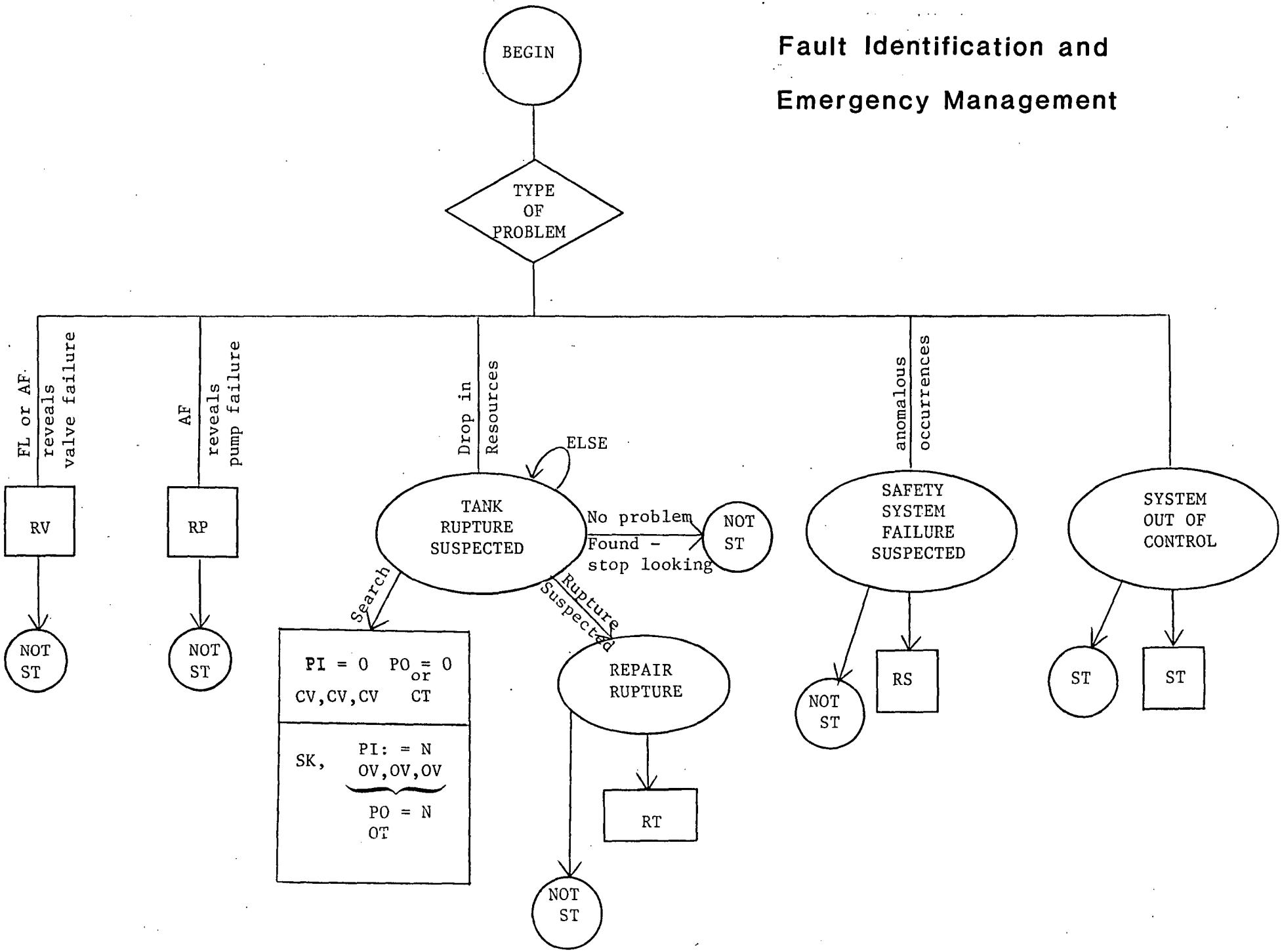
The rest of the subfunctions within the fault identification and emergency management function address requirements for fault detection and diagnosis. Repairs to valves and pumps are straightforward and are likely to occur as a result of routine valve checks conducted in steady-state management.

The procedure to detect and correct a tank rupture is not quite so straightforward. The procedure may be invoked because the operator

Figure 11

PLANT

Fault Identification and Emergency Management



observes an unexplained drop in system resources. The search procedure is laborious and may terminate in detecting a tank rupture, concluding that there is no tank rupture, or stopping an uncompleted search to undertake another control function.

The final fault identification subfunction is the identification and repair of the PLANT safety system. This is a little used procedure and is involved only when the operator has observed a critical number of anomalous system-initiated occurrences.

Uses of the PLANT Discrete Control Model

This model has several potential applications. It can be used either as an analysis tool to aid in understanding operator control performance or as a design tool to create a dynamic human-computer interface to control PLANT.

As an analysis tool, the model can be used to help explain operator control actions given system state. Its normative but nondeterministic form allows a great deal of flexibility in the sequence of control actions that are permitted within a control function or even within a subfunction. The structure is perhaps somewhat more people-oriented than KARL (Knaeuper, 1983), in that procedures have a beginning and an end; and once a procedure is initiated, it will not be preempted or terminated until a completion point has been reached. KARL, exhibiting one of the greatest strengths of computers, meticulously examines all system variables, updates its statistics every iteration, and assesses, given the new state, what it should be doing. As a result KARL could leave a procedure before completion and/or hop from procedure to procedure. People, given cognitive limitations, are much more likely, once beginning

a procedure, to continue its steps to conclusion, almost disregarding other symptoms arising in the intervening time. The discrete model has similar characteristics.

Although subject data has not been compared to the discrete control model, a logical next step in the model's validation is to compare it to human performance and evaluate its explanatory power. The model should have a high validity when compared to a "good" operator, i.e., one who had good output and faithfully followed prescribed procedures. The model should also be very helpful at identifying mismatches. The normative nature of the model suggests that these mismatches are likely to be operator mistakes.

As a design tool the discrete control model can be used to design an information display system which selects out, aggregates, or prioritizes system state information to facilitate the operator's control decisions depending on current operator control function and system state. The display system would have individual display pages that are tailored to the needs of operator control functions or subfunctions, as specified by the model. Used in this way, the discrete control model has potential utility as the basis of an on-line interactive decision aid. One strategy would be to let the operator specify the control function or subfunction currently underway, and given that function, the aid could prompt the operator with suggested next steps and activities as well as provide the required pieces of information.

An aid such as this may avoid some of the problems that KARL encountered when used as a human decision aid (Knaeuper and Morris, 1984). By allowing the operator to set the pace and having the program act as an aid rather than an expert, some of the problems of a nagging

on-line aid may be minimized. The interactive nature of an aid based on a discrete control model gives human operators much more control over the human-computer interface. This may be a desirable feature of an aid because as long as the human has the responsibility, s/he probably ought also to have the authority. The nondeterministic, heterarchic nature of discrete control models ensures, and in fact requires, this type of interface. The flexibility built into the interface requires that the human specify where s/he is and what the intent is. Yet the hierarchic structure of the model ensures that once the aid knows where the human wants to be in the control heterarchy, the appropriate information or procedural prompts can be provided.

REFERENCES

1. Knaeuper, A.E. A Model of Human Problem Solving in Dynamic Environments, M.S. Thesis, Center for Man-Machine Systems Research, Georgia Institute of Technology, Report No. 83-3, 1983.

2. Knaeuper, A.E. and Morris, N.M. "A Model-Based Approach for Online Aiding and Training in Process Control," Proceedings of the 1984 IEEE International Conference on Systems, Man, and Cybernetics, Halifax, Nova Scotia, 1984.

3. Miller, R.A. Identification of Finite State Models of a Human Operator, Final Report AFOSR Grant No. 77-3152, The Ohio State University Research Foundationn, 1979.

4. Miller, R.A. "A Systems Approach to Modeling Discrete Control Performance," in W.B. Rouse (ed.), Advance in Man-Machine Systems Research, Vol. II, JAI Press Inc., Greenwich, CO, 1985.

5. Mitchell, C.M. The Design of Computer-Based Integrated Information Displays, Ph.D. Thesis, The Ohio State University, 1980.

6. Morris, N.M. Human Problem Solving in Process Control, Ph.D. Thesis, Center for Man-Machine Systems Research, Georgia Institute of Technology, Report 83-2, 1983.